WHEN TO THINK DEEPLY: RESPONSE-CONDITIONED INHIBITORY ROUTING FOR LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Selective reasoning in language models must balance opportunities to correct an available answer against additional computation and harmful replacements. Inspired by inhibitory control in the human brain, we introduce Inhibitory Deliberation for Problem Reasoning (IDPR), which separates answer formation from answer release. A fast policy generates a candidate, and a response-conditioned gate either releases it or inhibits its release to invoke slow reasoning. Decomposed supervision uses paired fast–slow outcomes to learn slow-over-fast quality gain, generation cost, and corrective potential from the question, fast answer, and generation evidence. Calibration selects the most accurate of three scoring rules at fixed 8% coverage. On a fixed-policy benchmark of 5,000 mathematical problems, IDPR raises accuracy from 47.90% to 48.92% at an 8.20% slow-call rate, with 519.42 average generated tokens. The gain over Always-Fast is significant under a Holm-adjusted paired McNemar test ($p = 0.0012$), with 111 corrected errors and 60 harmful replacements. With response-and-evidence inputs and encoder family held fixed, IDPR-Matched's joint supervision-and-scoring scheme achieves 1.40 percentage points higher accuracy than RouteLLM-Q+R+E while reducing slow calls from 9.46% to 7.14% at their respective validation-calibrated thresholds.

1 INTRODUCTION

Reasoning-oriented language models can spend substantial test-time computation on multi-step solutions, verification, and self-correction (DeepSeek-AI, 2025; Muennighoff et al., 2025; Snell et al., 2025). Deploying these models requires allocating that computation across problems with different needs. In a fast-first system, a concise candidate answer is already available before additional reasoning is requested. The allocation decision can therefore use both the problem and the observed response, connecting inference-time computation to the choice of which answer to return.

Prior work targets different stages of compute allocation. Kimi k1.5, Open-Reasoner-Zero, and Skywork-OR1 strengthen reasoning through reinforcement learning (Kimi Team, 2025; Hu et al., 2025; He et al., 2025); s1 controls reasoning length (Muennighoff et al., 2025), and compute-optimal scaling adapts computation to prompt difficulty (Snell et al., 2025). These approaches improve reasoning policies rather than directly supervise post-answer replacement gain. Repeated sampling broadens solution coverage but requires reliable answer selection (Brown et al., 2024). Hybrid LLM and RouteLLM make query-only decisions using difficulty or preference prediction (Ding et al., 2024; Ong et al., 2025). FrugalGPT scores generated-answer quality within cost-constrained cascades (Chen et al., 2023). Closer to replacement decisions, learned deferral uses relative correctness or generation evidence (Jitkrittum et al., 2023; Gupta et al., 2024), while cascade routing updates quality and cost estimates using available outputs (Dekoninck et al., 2025). We study joint replacement-outcome supervision within this fast-first cascade setting.

Motivation. We seek to **separate answer formation from answer release**, so that additional reasoning is triggered by the predicted value of replacing the available answer. A slow call can correct a fast error, repeat it, or damage a correct answer; fast-answer confidence alone does not distinguish these outcomes. Inhibitory control in the human brain **suppresses ongoing or dominant responses to serve a goal** (Logan & Cowan, 1984; Miller & Cohen, 2001), while control allocation considers expected benefit and effort (Shenhav et al., 2013). Drawing on this principle, we propose Inhibitory

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Deliberation for Problem Reasoning (IDPR): a fast policy produces a candidate, and **a response-conditioned controller either releases it or inhibits its release to invoke a slow policy**. The controller learns **the consequences of replacing that particular candidate**, using paired fast–slow outcomes to supervise **signed quality gain, additional generation cost, and corrective potential**. The three prediction heads share a representation of the question, fast answer, and generation evidence. Validation selects the most accurate of three fixed-coverage scores and freezes its threshold.

We evaluate this design on 5,000 GSM8K and Mixture-of-Thoughts mathematical problems. Main IDPR reaches 48.92% accuracy at an 8.20% slow-call rate, compared with 47.90% for Always-Fast. Its slow calls correct 111 errors and damage 60 correct answers, yielding a net gain of 51. Three RouteLLM variants compare question-only and response-conditioned routing. With response-and-evidence inputs and encoder family fixed, IDPR-Matched's joint supervision-and-scoring scheme achieves 1.40 percentage points higher accuracy than RouteLLM-Q+R+E while reducing the slow-call rate from 9.46% to 7.14% at their respective validation-calibrated thresholds.

Our contributions are:

- A response-conditioned fast-first controller inspired by inhibitory control, releasing or withholding an available answer according to predicted replacement outcomes.
- Joint training on signed quality gain, slow generation cost, and corrective events, with fixed-coverage validation calibration linking predictions to the release-or-inhibit decision.
- A matched-input comparison showing a 1.40-point accuracy gain for the joint supervision-and-scoring scheme over preference-based routing at their respective validation-calibrated thresholds, alongside learned-router baselines, paired statistical tests, and analyses of corrections, harmful replacements, and threshold transfer.

2 Related Work

LLM routing and cascades. Routing methods differ in decision stage and score semantics. Hybrid LLM estimates query difficulty (Ding et al., 2024); RouteLLM learns paired-response preferences but routes from the query alone (Ong et al., 2025). FrugalGPT evaluates generated-answer quality to continue a cost-constrained cascade (Chen et al., 2023). Learned post-hoc deferral models relative outcomes: Diff-01 regresses the difference in two classifiers' correctness (Jitkrittum et al., 2023); LM cascades use token-level evidence with corrective-event or quality-difference targets (Gupta et al., 2024). Cascade routing integrates output-conditioned quality and cost estimates (Dekoninck et al., 2025). IDPR combines signed, format-aware quality gain, generation cost, and corrective supervision in a shared controller. Our RouteLLM adaptations add the fast answer (Q+R) and generation evidence (Q+R+E) to compare post-answer supervision-and-scoring schemes.

Test-time computation. Test-time computation determines how a reasoning policy uses its inference budget. Chain-of-thought prompting elicits intermediate steps (Wei et al., 2022), self-consistency aggregates sampled answers (Wang et al., 2023), and Tree of Thoughts explores and evaluates alternative paths (Yao et al., 2023). Reinforcement learning strengthens reasoning policies (DeepSeek-AI, 2025; Kimi Team, 2025; Hu et al., 2025; He et al., 2025), while s1 controls reasoning length (Muennighoff et al., 2025) and compute-optimal scaling allocates computation according to problem difficulty (Snell et al., 2025). Sampling increases solution coverage but still requires reliable answer selection (**?**). These approaches concern reasoning computation; IDPR decides whether to invoke it after observing a fast answer.

Uncertainty and response-conditioned control. Generation likelihood and logit margins describe uncertainty, but replacement value also depends on the slow policy: it may correct an error, repeat it, or damage a correct answer. Paired outcomes supervise these transitions. Dual-process and cognitive-control accounts distinguish default responses from controlled processing (Evans & Stanovich, 2013; Kahneman, 2011; Botvinick et al., 2001; Miller & Cohen, 2001). Inhibitory control suppresses initiated responses (Logan & Cowan, 1984), while effort and expected-value accounts relate control allocation to anticipated returns and cognitive cost (Kool et al., 2010; Westbrook et al., 2013; Shenhav et al., 2013). These ideas motivate IDPR's separation of answer formation from release, with intervention learned from fast–slow outcomes.

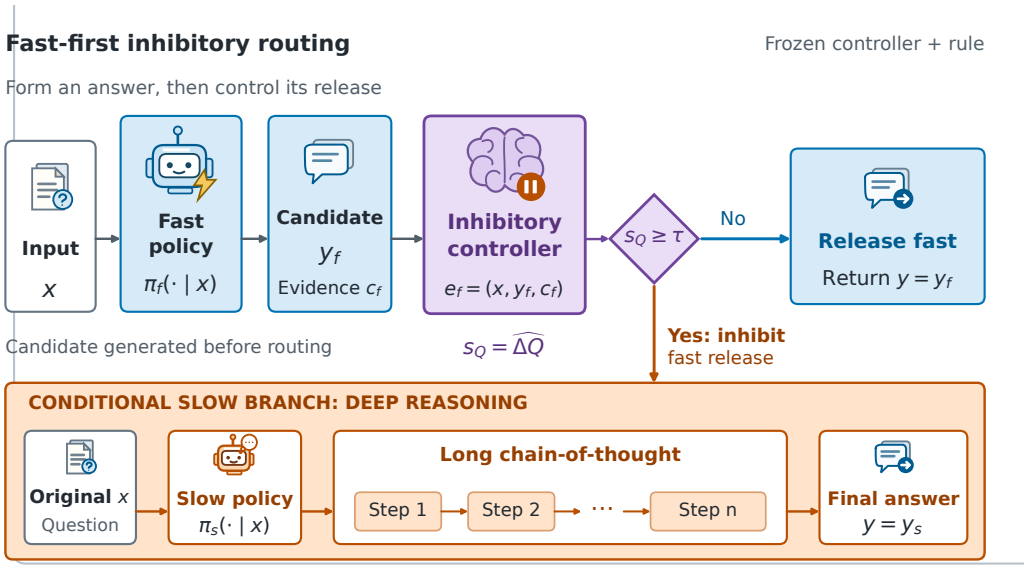


Figure 1: Online execution of IDPR. The fast policy forms a candidate y_f and evidence c_f. The inhibitory controller uses predicted quality gain $s_Q = \widehat{\Delta Q}$ and a frozen threshold τ to release y_f or activate the orange slow branch. That branch receives the original question x and generates a multi-step solution followed by its final answer. Numbered boxes depict a variable-length trace.

Table 2 in Appendix A.5 compares routing approaches by decision stage, input, and supervision.

3 Method

IDPR learns when to replace a realized fast answer. We select for returned-answer accuracy at fixed calibration coverage. Paired outcomes train a quality-ranking surrogate with cost and correction auxiliaries; separate calibration chooses the score and boundary (Figure 2). Online, the fast candidate and generation evidence determine release or inhibition (Figure 1). The policies, controller, and rule remain fixed at deployment; utility summarizes the resulting cost–quality trade-off.

3.1 Fast-first selective deliberation

We fix two generation policies: π_f produces a concise answer; π_s executes a reasoning protocol with fixed prompts, decoding, stopping, and conditional fallback and format repair (Appendix A.1). One slow call denotes activation of this complete protocol. For a problem x, routing begins after generating the fast candidate

$$y_f \sim \pi_f(\cdot \mid x). \tag{1}$$

The response y_f is the default final answer. During its generation, we collect signals c_f describing confidence, output structure, and computational behavior. The state (x, y_f, c_f) lets the controller assess replacement value before committing to the candidate's release.

The controller chooses a route $r \in \{\text{fast}, \text{slow}\}$ before releasing the answer. Choosing the fast route returns y_f directly. Choosing the slow route inhibits its release and invokes $y_s \sim \pi_s(\cdot \mid x)$; the resulting output becomes the final response:

$$y = \begin{cases} y_f, & r = \text{fast}, \\ y_s, & r = \text{slow}. \end{cases} \tag{2}$$

The slow policy receives the original problem x, rather than editing y_f. Controller inputs require no gold answer and precede the slow call, so routing predicts an unobserved replacement outcome. Figure 1 confines deep reasoning to the slow branch.

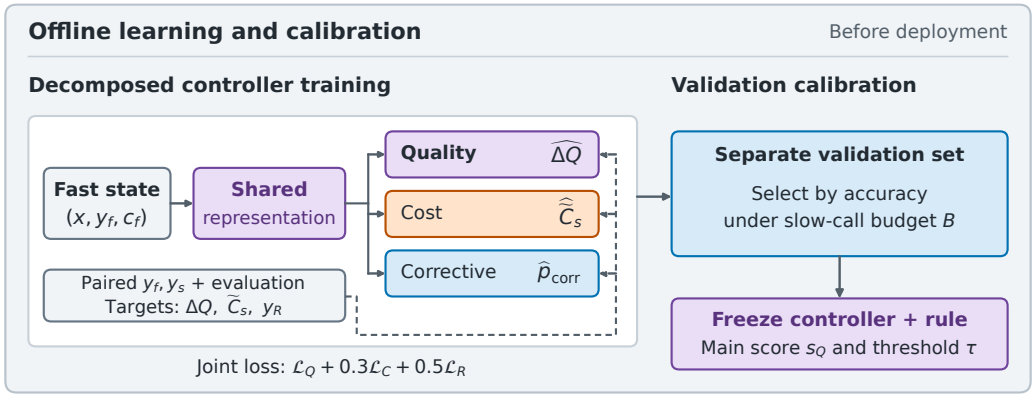


Figure 2: Offline preparation of IDPR. Paired fast–slow outcomes supervise quality, normalized-cost, and corrective heads sharing a fast-state representation; dashed arrows denote supervision. After fitting and checkpoint selection, separate validation compares routing scores by accuracy at fixed coverage B. The controller and selected rule are frozen for online execution (Figure 1). Main routing uses s_Q, with auxiliary cost and correction losses.

3.2 Response-conditioned evidence and encoding

The fast-side state $e_f = (x, y_f, c_f)$ has textual and numerical views. Text includes the question, parsed fast answer, source tag, parsing status, and answer-form tag. Numerical evidence describes confidence, parseability, generation cost, degeneracy, and source. Paired fast–slow outcomes supervise replacement-value prediction from these views.

The canonical controller encodes text with DistilBERT and masked-mean pooling over non-padding tokens. A $14 \to 256 \to 256$ MLP embeds the standardized numerical features. Concatenating the text and numerical representations feeds a two-layer fusion MLP with 256-dimensional hidden and output representations. Both MLPs use GELU and dropout 0.1 after the first activation. Three scalar heads predict signed quality gain, normalized slow cost, and a corrective-event score. These parallel heads yield alternative routing scores from one shared representation. Appendix A.2 specifies the feature transformations and input serialization.

The main system uses the canonical 14-dimensional controller; the matched comparison uses a separate XLM-R controller with source-free, 16-dimensional evidence (Section 4).

3.3 Learning from paired replacement outcomes

Training pairs fast and slow outputs for the same problem. Gold-answer evaluation, format status, and generated-token counts define three targets: signed quality gain, additional slow cost, and a corrective event. Training thus observes both routing outcomes, while controller inputs contain only the fast-side state e_f. The corrective loss separately emphasizes infrequent fast-error-to-slow-success transitions. Figure 2 links joint supervision to validation calibration.

Quality gain. Let $A_f, A_s \in \{0, 1\}$ indicate whether the fast and slow answers are evaluated as correct, and let F_f, F_s denote their format penalties. The quality head predicts the difference

$$\Delta Q = (A_s - A_f) - \rho(F_s - F_f), \qquad \rho = 0.25, \tag{3}$$

where $F \in \{0, 0.25, 1\}$ depends on answer extraction, truncation, and degeneracy. With equal penalties, correction, harmful replacement, and unchanged correctness yield $+1$, -1, and zero. The format term distinguishes output validity within correctness ties. This format-aware replacement target supplies a ranking surrogate, while calibration evaluates returned-answer accuracy. It models the change from fast to slow, rather than fast-answer uncertainty alone.

Slow generation cost. The cost head supplies auxiliary supervision for additional generation and supports cost-aware scoring. Let C_s count tokens across the activated slow protocol. Its normalized

target is

$$\widetilde{C}_s = \frac{\log(1 + C_s)}{\log(1 + C_{\text{ref}})}, \qquad C_{\text{ref}} = 14{,}400. \tag{4}$$

The logarithm compresses differences among long traces; the fixed reference sets the regression scale. This target measures incremental slow cost because fast generation has already occurred. The optional cost-aware score uses the token-scale estimate $\widehat{C}_s$, whereas main routing uses quality alone. Appendix A.1 specifies inversion and prediction-side clipping.

Corrective potential. The auxiliary corrective head assigns a score to a specific transition: an explicitly incorrect fast answer becoming a correct slow answer. Its binary training label is

$$y_R = \mathbf{1}[\text{fast status is incorrect } \wedge\ A_s = 1]. \tag{5}$$

Unknown fast-answer status, harmful replacements, and unchanged outcomes are excluded from the positive class. Quality regression captures signed consequences; the auxiliary classification loss instead emphasizes correction. Class weighting and sampling emphasize rare corrective examples.

Joint training. We fit the three heads with weighted Smooth-L_1 losses for quality and cost and weighted binary cross-entropy for correction:

$$\mathcal{L} = \mathcal{L}_Q + 0.3\mathcal{L}_C + 0.5\mathcal{L}_R. \tag{6}$$

All three losses update shared features; cost and correction are auxiliary to quality-only routing. Weighting and sampling emphasize corrective examples. Outputs rank examples under that training distribution. Appendix A.1 gives weights, sampling, batch-mean losses, and checkpoint selection.

Controller fitting, checkpoint selection on internal validation, and operating-point calibration use separate data roles (Appendix A.1). Section 5.2 compares the complete decomposed-supervision and preference-supervision schemes, including their respective routing scores.

3.4 Routing scores and coverage calibration

The learned outputs support alternative rankings of the available fast answers. We consider predicted quality gain s_Q, a corrective-event score s_R, and an optional cost-aware score s_U:

$$s_Q = \widehat{\Delta Q}, \qquad s_R = \widehat{p}_{\text{corr}}, \qquad s_U = \widehat{\Delta Q} - \lambda_{\text{inf}}\widehat{C}_s. \tag{7}$$

Here $\widehat{C}_s$ is predicted slow cost in tokens, $\widehat{p}_{\text{corr}}$ is a sigmoid ranking score, and λ_{inf} sets cost sensitivity. The controller switches when $s(e_f) \geq \tau$ and otherwise retains the fast answer. Calibration may specify an identifier tie-break; raising the threshold narrows eligibility. We set the threshold by calibration coverage rather than a zero-gain cutoff on these learned scores.

Calibration completes offline preparation (Figure 2) with the fitted controller fixed. The candidate scores are $\mathcal{S}_B = \{s_Q, s_R, s_U\}$, using $\lambda_{\text{inf}} = 2 \times 10^{-5}$ and common calibration coverage $B = 8\%$. For each score, we stably sort $n = 2{,}000$ separate validation examples in descending order, preserving input order within ties, and select the top $k = 160$. This defines selection $D_s = (d_{s,1}, \ldots, d_{s,n})$ and boundary $\tau(s)$ at the kth score, with $d_{s,i} = 1$ for a selected example and 0 otherwise. Using the fast and slow correctness indicators $A_{f,i}$ and $A_{s,i}$, we evaluate

$$\begin{aligned} \text{Acc}_{\text{val}}(D_s) &= \frac{1}{n}\sum_{i=1}^{n}\big[(1 - d_{s,i})A_{f,i} + d_{s,i}A_{s,i}\big], \\ \text{SlowRate}_{\text{val}}(D_s) &= \frac{1}{n}\sum_{i=1}^{n} d_{s,i}. \end{aligned} \tag{8}$$

Accuracy is computed on the returned answers, while slow-call rate measures the selected fraction. We choose among these fixed-coverage candidates:

$$s^* = \arg\max_{s \in \mathcal{S}_B} \text{Acc}_{\text{val}}(D_s), \qquad \tau^* = \tau(s^*). \tag{9}$$

Accuracy selects s_Q with $\tau^* = -0.02437508851289749$; ties are resolved by higher utility, then lower average token cost. Cost and correction remain auxiliary training targets, while quality determines inhibition. This negative boundary implements the selected calibration coverage.

The canonical test rule applies $s_Q \geq \tau^*$ without reselecting a top-k. Calibration uses exact top-k selections; a transferred scalar boundary can include additional ties. The four learned comparators calibrate prespecified scores to 160 examples without correctness labels. IDPR-Matched also freezes a UID tie-break (Appendix A.1). Test call rates reflect score distributions and boundary ties.

For descriptive cost–quality reporting and validation tie-breaking, we use

$$U = A - \lambda_{\text{tok}} C - \rho F, \qquad \lambda_{\text{tok}} = 2 \times 10^{-5}, \quad \rho = 0.25. \tag{10}$$

Here A and F refer to the returned answer, and $C = C_f + \mathbf{1}[r = \text{slow}]C_s$ counts both the initial fast generation and any selected slow generation. The coefficient λ_{tok} defines the reporting metric, whereas λ_{inf} governs the optional routing score in Equation 7. Fast generation is shared by both routing choices, so its cost cancels when comparing slow and fast utility, as derived in Appendix A.3.

4 Experimental Setup

Data and policies. We use GSM8K (Cobbe et al., 2021) and mathematical subsets of Mixture-of-Thoughts (Face, 2025). We split 2,500 paired fast–slow examples into 2,000 training, 250 internal-validation, and 250 internal-test examples. Separate calibration (2,000) and benchmark (5,000) sets are each source-balanced. The benchmark is disjoint from controller training and calibration but was used in earlier system development. The fixed pair comprises an answer-only fast policy based on Qwen2.5-Math-7B (Yang et al., 2024) and an OpenR1-Distill-7B (Open R1 Team, 2025) slow policy, with greedy fast decoding and sampled slow reasoning (Appendix A.1).

Compared methods. Always-Fast returns every fast answer. Random Same-k, Low Avg LogProb Same-k, and Low Logit Gap Same-k match main IDPR's 410 slow calls, not token costs. Our XLM-R RouteLLM adaptations learn three classes (slow-win, tie, fast-win) and rank by slow-win probability. They use the question (Q), add the parsed fast answer (Q+R), or also source-free, 16-dimensional evidence (Q+R+E; Appendix A.2). IDPR-Matched shares Q+R+E inputs, encoder family, and splits, with decomposed supervision (Section 3) and quality-gain ranking. Direct Qwen2.5-Math-7B-Instruct is a separate base-policy reference, outside this fixed-pair comparison.

Training and evaluation protocol. Preprocessing is fitted on training data. Internal validation selects RouteLLM checkpoints by highest slow-win AUPRC and IDPR-Matched by lowest blended multi-task loss; results use selected runs, not seed averages. Calibration selects the main score (Equation 9) and sets decision boundaries. Models, preprocessing, and calibrated routing rules are frozen before test scoring; test selections are fixed before completing the shared slow-output cache. All methods share fast outputs and identifiers. Appendix A.1 details the protocol.

Metrics and cost accounting. We report accuracy, its percentage-point change from Always-Fast, slow-call rate, average generated tokens, and utility. Corrective precision counts explicitly incorrect fast answers replaced by correct slow answers, excluding unknown fast status; harmful precision counts correct fast answers replaced by incorrect or unknown slow answers. Both divide by selected counts. Uncertainty estimates and tests use paired identifiers (Appendix A.4). Token costs include fast generation and all recorded initial, fallback, and repair tokens within selected slow invocations, including replaced attempts. The question-only router uses the same accounting. Prompt tokens, controller computation, and failures or reruns outside stored invocations are excluded.

5 Results

5.1 Main results

Main IDPR improves accuracy from 47.90% for Always-Fast to 48.92%, a gain of 1.02 percentage points at an 8.20% slow-call rate (Table 1). This gain requires 519.42 average generated tokens per example, compared with 5.26 for Always-Fast. Utility decreases from 0.4789 to 0.4731: the accuracy-first operating point trades additional generation for more correct answers.

At 410 slow calls, Random Same-k lowers accuracy to 46.76%, while Low Avg LogProb, the strongest confidence baseline by accuracy, reaches 48.22%. Main IDPR has higher observed accuracy and corrective precision while using fewer tokens than Low Avg LogProb (Table 1).

Table 1: Results on 5,000 examples. Same-k methods make 410 slow calls; learned routers transfer validation-calibrated thresholds. Accuracy and corrective precision (CP) are percentages; Δ is the change from Always-Fast in percentage points. Tokens are average generated tokens per example.

Method	Slow rate	Accuracy	Δ	Utility	Tokens	CP
Always-Fast	0.00%	47.90	–	**0.4789**	**5.26**	–
Random Same-k	8.20%	46.76	-1.14	0.4517	517.91	13.41
Low Avg LogProb Same-k	8.20%	48.22	$+0.32$	0.4645	582.68	21.22
Low Logit Gap Same-k	8.20%	47.84	-0.06	0.4571	714.73	11.46
RouteLLM-Q	7.24%	47.76	-0.14	0.4635	456.45	20.99
RouteLLM-Q+R	9.26%	47.36	-0.54	0.4574	501.22	19.22
RouteLLM-Q+R+E	9.46%	46.74	-1.16	0.4525	451.53	16.91
IDPR-Matched	7.14%	48.14	$+0.24$	0.4690	383.76	23.53
IDPR	8.20%	**48.92**	$\mathbf{+1.02}$	0.4731	519.42	**27.07**

Direct-model reference. The direct Qwen2.5-Math-7B-Instruct reference reaches 53.32% raw accuracy, 386.99 average tokens, utility 0.4177, and 21.92% unknown/unparseable outcomes under the same parser and cost definition. It is more accurate and uses fewer tokens than main IDPR, but has lower utility under the format penalty. Policy choice sets available answers and costs; Table 1 instead compares routing within a fixed policy pair.

5.2 Response inputs and training objectives

Under the reported training and calibration, the response-conditioned Q+R and Q+R+E variants have lower accuracy than Q despite more slow calls (Table 1).

With Q+R+E inputs and encoder family held fixed, IDPR-Matched gains 1.40 percentage points over RouteLLM-Q+R+E while reducing slow calls from 9.46% to 7.14% (Table 1). This compares joint supervision-and-scoring schemes at separately calibrated thresholds, not a same-k ranking.

Main IDPR has 0.78 percentage points higher accuracy than IDPR-Matched and uses 135.66 more average tokens. The accuracy difference is not significant after Holm correction ($p = 0.1083$). Encoder and evidence representations differ (Section 4), so complete configurations are compared.

5.3 Paired uncertainty and significance

After Holm correction, main IDPR's accuracy gains over Always-Fast and all three RouteLLM variants are significant (Figure 3(a)). The gain over Always-Fast has a 95% bootstrap interval of $[0.50, 1.52]$ percentage points and an adjusted McNemar p-value of 0.001182. The 1.40-point gain for IDPR-Matched over Q+R+E is also significant (adjusted $p = 0.0001593$; interval $[0.80, 2.02]$). Main IDPR's corrected differences from Low Avg LogProb and IDPR-Matched are nonsignificant.

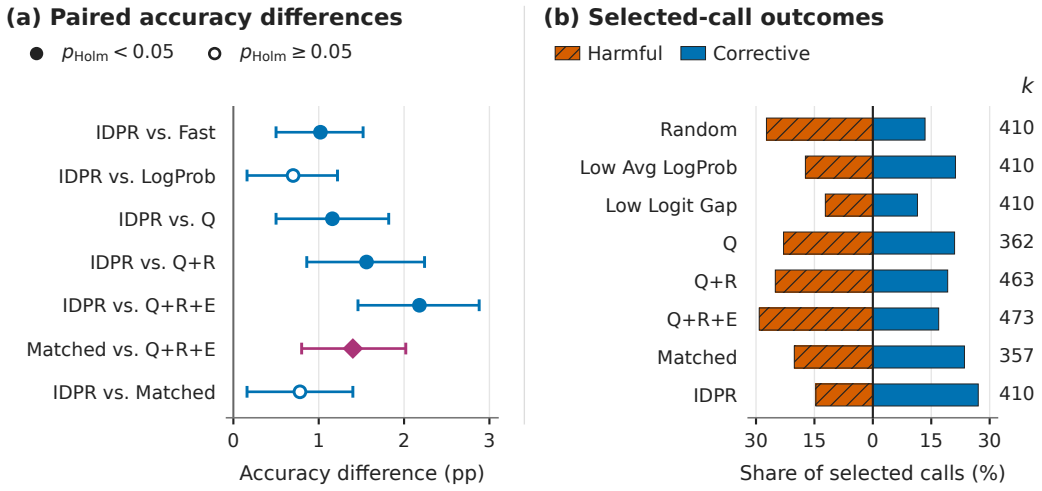
Bootstrap intervals quantify per-comparison uncertainty with model outputs held fixed, whereas significance follows the Holm-adjusted tests. Thus, the 0.70-point difference from Low Avg LogProb has a bootstrap interval of $[0.16, 1.22]$, yet its adjusted test is not significant ($p = 0.1064$). These intervals exclude variability from controller retraining and generation resampling.

5.4 Selection outcomes

To connect accuracy differences to replacement outcomes, let G count selected examples whose correctness indicator changes from 0 to 1, and H count those changing from 1 to 0. For N benchmark examples, the change relative to Always-Fast is

$$\Delta \,\mathrm{Acc} = \frac{G - H}{N}. \tag{11}$$

Corrective and harmful precision in Figure 3(b) use selected count k, whereas Equation 11 uses all N examples. Strict correction requires an explicitly incorrect fast answer; recoveries from unknown fast status contribute to G but not to strict corrective counts or the positive training class.



Fast = Always-Fast; LogProb = Low Avg LogProb; Q variants = RouteLLM.
Matched = IDPR-Matched. Intervals: unadjusted 95% paired bootstrap CIs.
Tests: Holm-adjusted exact McNemar (14 comparisons); 5,000 test examples.

Figure 3: (a) Accuracy differences have unadjusted 95% paired bootstrap intervals; filled/open markers indicate significant/nonsignificant Holm-adjusted exact McNemar tests. (b) Strict corrective (right) and harmful (left) percentages use each method's selected-call denominator k; both are nonnegative. The first three controls use $k = 410$. Tables 3 and 4 give complete numerical results.

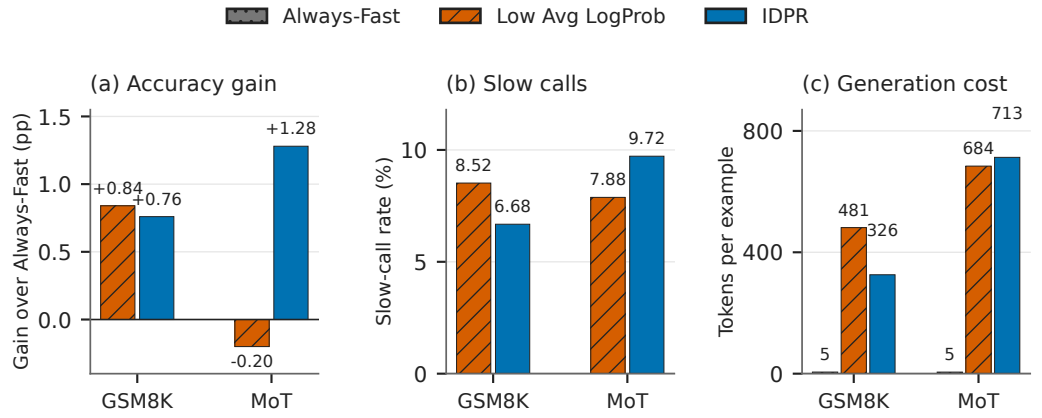


Figure 4: Source-wise gains, allocation, and generation cost on 2,500 examples per source. (a) Accuracy gain over the source-specific Always-Fast baseline. (b) Slow-call rate within each source. (c) Mean generated tokens, including Always-Fast; token labels are rounded to whole tokens. Always-Fast has zero gain and zero slow calls. Exact values appear in Table 5.

Main IDPR gains 51 correct answers (1.02 points) from 111 corrections and 60 harmful switches. At the same 410 calls, Low Avg LogProb gains 16 from 87 corrections and 71 harmful switches.

Relative to Q+R+E, IDPR-Matched raises corrections from 80 to 84 and reduces harmful switches from 138 to 72, despite 116 fewer calls. The 70-answer gain is measured over selections at different call rates, with higher corrective precision and a lower harmful-switch proportion.

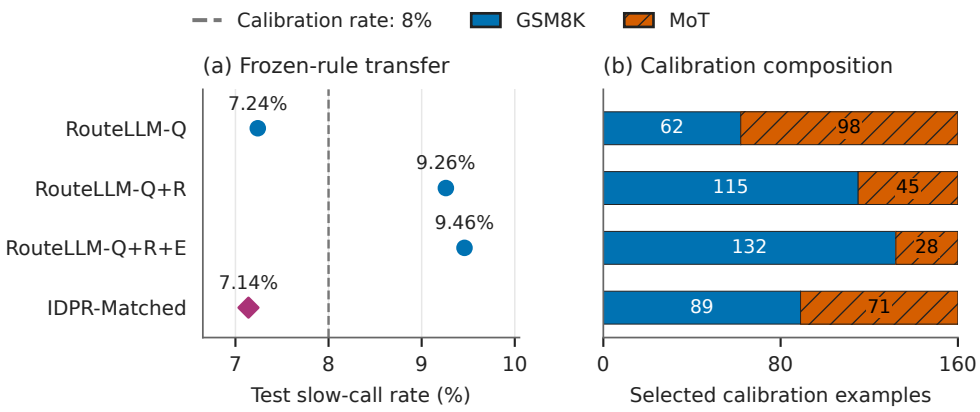


Figure 5: (a) Frozen-rule slow-call rates on 5,000 test examples are compared with 8% calibration coverage (dashed). (b) Bars show source counts for each router's 160 calibration selections. IDPR-Matched transfers its threshold and identifier tie-break. Table 7 lists thresholds and test counts.

6 Analysis

6.1 Source-wise behavior

IDPR improves accuracy over Always-Fast on both sources, gaining 0.76 points on GSM8K and 1.28 on MoT (Figure 4). Low Avg LogProb is 0.08 points more accurate on GSM8K but uses more tokens; it loses 0.20 points on MoT. IDPR allocates more slow calls to MoT, where the fast policy is less accurate. Call counts match Low Avg LogProb only overall, not within sources, so comparisons combine ranking and allocation differences. Because source metadata enters the canonical controller, these patterns describe routing within the evaluated mixture, not unseen-source transfer.

Additional comparisons of controller and fast-policy configurations appear in Appendix A.6.

6.2 Threshold transfer

At the common 8% calibration rate, Q selects more MoT examples, while Q+R, Q+R+E, and IDPR-Matched select more GSM8K examples (Figure 5). Frozen-rule transfer yields test slow-call rates of 7.14%–9.46%, rather than a fixed test call count. These results concern the benchmark mixture; Appendix A.6 compares the main and earlier cost-aware configurations.

6.3 Replacement quality and generation cost

The matched joint supervision-and-scoring scheme gains accuracy over Q+R+E mainly through fewer harmful switches (Figure 3(b)). Replacement quality involves correcting errors and preserving correct answers. Signed targets distinguish correction from harm; corrective labels mark corrections.

Equal slow-call counts do not imply equal token costs because slow traces vary in length. At 410 calls, main IDPR uses fewer tokens than the confidence baselines; Q+R+E makes more calls but uses fewer tokens than the main IDPR configuration (Table 1).

7 Conclusion

IDPR draws on inhibitory control to separate answer formation from release, with paired replacement supervision and coverage-calibrated routing. On 5,000 mathematical problems, main IDPR gains 1.02 percentage points over Always-Fast at an 8.20% slow-call rate. With response-and-evidence inputs and encoder family held fixed, the matched joint supervision-and-scoring scheme gains 1.40 points over preference routing at separately calibrated thresholds. Evaluation on unseen sources and additional model pairs would test transfer beyond mathematical reasoning.

REPRODUCIBILITY STATEMENT

Section 3 describes the controller representation, training targets, and routing rule; Appendix A.1 records the reported training and calibration settings, and Appendix A.4 specifies the statistical procedure. The evaluation freezes test views, model scores, selections, cached slow outputs, and file hashes before computing final metrics. Appendix A.7 preserves the numerical results underlying the experimental figures. The supplementary artifact is organized around preprocessing and evaluation scripts, configuration files, frozen identifiers, and table-generation commands.

AI USE STATEMENT

Generative AI tools assisted with language editing, structural revision, LaTeX formatting, and plotting code for the reported experimental summaries. Authors are responsible for checking AI revisions and verifying all claims, equations, citations, experimental results, and the final submission.

REFERENCES

Matthew M. Botvinick, Todd S. Braver, Deanna M. Barch, Cameron S. Carter, and Jonathan D. Cohen. Conflict monitoring and cognitive control. *Psychological Review*, 108(3):624–652, 2001.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12987–13010. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/dekoninck25a.html.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, 2024.

Jonathan St. B. T. Evans and Keith E. Stanovich. Dual-process theories of higher cognition: Advancing the debate. *Perspectives on Psychological Science*, 8(3):223–241, 2013.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KgaBScZ4VI.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Cheng Peng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, et al. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Wittawat Jitkrittum, Neha Gupta, Aditya K Menon, Harikrishna Narasimhan, Ankit Rawat, and Sanjiv Kumar. When does confidence-based cascade deferral suffice? In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 9891–9906. Curran Associates, Inc., 2023. doi: 10.52202/075280-0431. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1f09e1ee5035a4c3fe38a5681cae5815-Paper-Conference.pdf.

Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Wouter Kool, Joseph T. McGuire, Zev B. Rosen, and Matthew M. Botvinick. Decision making and the avoidance of cognitive demand. *Journal of Experimental Psychology: General*, 139(4): 665–682, 2010.

Gordon D. Logan and William B. Cowan. On the ability to inhibit thought and action: A theory of an act of control. *Psychological Review*, 91(3):295–327, 1984. doi: 10.1037/0033-295X.91.3.295.

Earl K. Miller and Jonathan D. Cohen. An integrative theory of prefrontal cortex function. *Annual Review of Neuroscience*, 24:167–202, 2001.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. In *International Conference on Learning Representations*, 2025.

Open R1 Team. Openr1-distill-7b. https://huggingface.co/open-r1/OpenR1-Distill-7B, 2025.

Amitai Shenhav, Matthew M. Botvinick, and Jonathan D. Cohen. The expected value of control: An integrative theory of anterior cingulate cortex function. *Neuron*, 79(2):217–240, 2013.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Andrew Westbrook, Daria Kester, and Todd S. Braver. What is the subjective cost of cognitive effort? load, trait, and aging effects revealed by economic preference. *PLOS ONE*, 8(7):e68210, 2013.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

A Supplementary Material

A.1 Controller training and selection

The paired controller data contain 2,000 training, 250 internal-validation, and 250 internal-test examples. Training outcomes in the 2,000-example partition comprise 277 slow wins (fast incorrect and slow correct), 1,497 ties, and 226 fast wins. The total loss weights are $(1.0, 0.3, 0.5)$ for quality, cost, and correction. The corrective positive-class weight is $\sqrt{1723/277} = 2.4940362079364835$, with a positive sampling multiplier of 3. The canonical run completes 10 epochs and selects its epoch-4 checkpoint using the lowest internal-validation blended loss.

Data provenance. The source-balanced benchmark is a custom 5,000-example evaluation, including 2,500 GSM8K-sourced records, rather than the standard GSM8K test split. Its separation from controller fitting and calibration concerns those partitions. OpenR1-Distill-7B was fine-tuned on Mixture-of-Thoughts (Open R1 Team, 2025); the extent of overlap between this benchmark and that training corpus is unknown. Determining this overlap requires the original source splits, sampling and deduplication manifests, and model/data revisions.

Loss reduction and checkpoint selection. Let m be the current batch size and $w_{Q,i}$, $w_{C,i}$, and $w_{R,i}$ the per-example weights read from the training records. The canonical trainer computes

$$\begin{aligned}
\mathcal{L}_Q &= \frac{1}{m}\sum_{i=1}^{m} w_{Q,i}\,\ell_1(\widehat{\Delta Q}_i, \Delta Q_i),\\
\mathcal{L}_C &= \frac{1}{m}\sum_{i=1}^{m} w_{C,i}\,\ell_1(\widehat{\widetilde{C}}_{s,i}, \widetilde{C}_{s,i}),\\
\mathcal{L}_R &= -\frac{1}{m}\sum_{i=1}^{m} w_{R,i}\big[p_+ y_{R,i}\log\sigma(z_i) + (1-y_{R,i})\log(1-\sigma(z_i))\big].
\end{aligned} \tag{12}$$

Here ℓ_1 is Smooth-L_1 with its default transition parameter $\beta = 1$, z_i is the corrective logit, and $p_+ = 2.4940362079364835$. The implementation uses binary cross-entropy with logits for numerical stability. It averages weighted losses over examples, not over total weight. Sampling is with replacement, with weights 3 for corrective positives and 1 for other examples and 2,000 draws per epoch. The training implementation uses AdamW with separate parameter groups for trainable encoder and head parameters, followed by linear warmup and decay. Internal-validation selection combines mean unweighted quality Smooth-L_1, cost Smooth-L_1, and corrective BCE with coefficients 1, 0.3, and 0.5, omitting training sample weights and the corrective positive-class weight.

Canonical score calibration. Equation 8 defines returned-answer accuracy and coverage for each candidate selection; Equation 9 selects the operating rule.

The separate 2,000-example calibration set and 5,000-example test set each contain equal numbers of GSM8K and MoT examples. The main run uses only the 8% coverage setting, producing one top-160 candidate for each of cost-aware, quality-gain, and corrective-event scores. Their validation accuracies are 46.80%, 46.95%, and 46.80%, respectively. The selected quality-gain threshold is -0.02437508851289749; applying $s_Q \geq \tau$ on test makes 410 calls. Validation uses stable top-k selection, while test uses the frozen scalar comparator. The stored A/B/C schemes select the same candidate in this run, not three independently evaluated budgets.

The algebraic inverse of Equation 4 is $C_s = \exp\big(\widetilde{C}_s \log(1+C_{\text{ref}})\big) - 1$. Deployment clips normalized predictions to $[0, 1]$, applies this inverse, and caps the result at 14,400 tokens. The canonical training-validation diagnostic instead clips the prediction to $[0, 1.5]$ before inversion. These prediction-side transformations are distinct from training-target construction. Cost-aware diagnostic rankings determine neither main quality-only routing nor loss-based checkpoint selection.

For the RouteLLM adaptations, the three preference classes are slow-win, tie, and fast-win. The sequence classifier uses the XLM-R family and the frozen artifacts below.

Artifact	Frozen identifier
RouteLLM commit	`0b64fdafe049e596a3f5657c219329f24af24198`
Checkpoint	`routellm/bert_gpt4_augmented`
Checkpoint revision	`86237e3df400762178ea98379477b8296e66d5e4`

All variants use seeds 13, 42, and 2025. Q selects seed 42 at epoch 6, Q+R selects seed 2025 at epoch 3, and Q+R+E selects seed 42 at epoch 4 by internal-validation slow-win AUPRC. They are independently calibrated to 160 slow calls on the 2,000-example calibration set, with thresholds 0.689285, 0.473572, and 0.611396. IDPR-Matched selects seed 2025 at epoch 5 by minimum internal-validation blended loss, with the same unweighted validation components and coefficients described above; its selected loss is 0.6260585652664303. Across equal-loss seed candidates, the smaller numeric seed wins. The four routers' prespecified scores are calibrated without correctness labels. On test, their frozen rules select 362, 463, 473, and 357 examples, respectively.

Matched boundary ties. IDPR-Matched ranks predicted quality gain in descending order and breaks ties by ascending lexicographic order of the pair UID. The paper-main manifest fixes $\tau_M = 0.7578125$ and the boundary UID

`07739c5484c40e655217bf17`.

A slow call is selected when the score exceeds τ_M, or when it equals τ_M and its UID is no larger than this boundary. The 160th and 161st calibration scores both equal τ_M, so the UID rule is required to obtain the recorded 160 selections (89 GSM8K and 71 MoT). The identifier is the first 24 hexadecimal characters of a SHA-256 digest built from the normalized source, source UID, and normalized-question hash using the pair-UID serialization in the evaluation code. The rule uses this pair UID, not the raw source UID or row index, and is transferred unchanged to test.

Baseline replay. Random Same-k initializes a Python random generator with the main evaluation seed plus 999, giving 1041. The original A/B/C scheme loop consumes this generator sequentially; the main B selection is replayed with the same scheme and example ordering, rather than sampled from a newly initialized generator. Confidence rankings are stable ascending sorts, so tied scores preserve input order. The original evaluator supplies a zero fallback for missing confidence values; the frozen benchmark reconstruction reads the stored numerical scores directly. This differs from the controller's numerical-feature imputation described in Appendix A.2.

Generation and cache accounting. The main evaluation records seed 42, chat-template formatting, bfloat16 execution, and eager attention. Fast generation uses one beam, no sampling, a 1,024-token prompt truncation limit, and at most 64 new tokens. Slow generation samples with temperature 0.6 and top-p 0.95, initially allowing 6,144 new tokens. It checks for a parsable final answer every eight generated tokens using a 512-token window. Online stopping accepts explicit final-answer lines, boxed answers, or GSM-style answer markers, with last-answer-like fallback disabled. Parser statuses corresponding to no final answer, multiple-answer drift, truncation without an answer, or repetitive degeneration trigger a fresh generation from the original problem prompt with an 8,192-token limit. If the resulting status remains in this set, a greedy format-repair call receives the question and latest response and generates at most 64 tokens. Each triggered stage replaces the prior candidate without correctness-based selection. Slow-token counts include every recorded attempt in the invocation, even if its output is replaced.

The system prompts are fixed as follows. Fast: "You are a fast intuitive solver. Do NOT show reasoning. Output exactly one line in the format: Final: <answer>". Slow: "You are a careful reasoner. Solve the problem step by step. At the end, output ONE single line in the exact format: Final: <answer>." Repair: "You are a format enforcer. Output exactly ONE line: Final: <answer>." Direct Qwen2.5-Math-7B-Instruct has a 512-token cap and is not this pair's always-slow run.

Frozen selections require 1,994 distinct slow outputs: 1,477 are cached, and 517 missing outputs are generated. Cache completion is separate from fallback and format repair within a slow invocation. For each selected example, token totals add fast generation to the accumulated slow tokens recorded across that invocation's initial, fallback, and repair attempts, including attempts whose text is subsequently replaced. The question-only router follows the same fast-first accounting. Prompt tokens, controller

computation, and failures or reruns outside the stored invocation are excluded. Metrics follow checks of identifiers, source counts, selections, cache coverage, and file hashes.

A.2 Fast-side evidence and parsing

The canonical controller uses five evidence groups: confidence, parseability, generation cost, degeneracy, and source. Confidence includes average token log-probability and average logit gap. Parseability includes the number of candidate answers, parsed answer form, parse status, and empty-answer flag. Generation cost includes fast-token count, latency, and throughput. Degeneracy includes repetition and truncation. Source is represented both in the textual view and by numerical indicators. All evidence is available before routing and does not use the gold answer.

The canonical controller's 14 standardized numerical features are average log-probability, average logit gap, log fast-token count, log latency, log throughput, repetition score, candidate count, log answer length, parse-success, malformed-output, truncation, empty-answer, and two source indicators. Section 3.2 describes their encoding and fusion with the text representation.

The text serialization places the question first, followed by the parsed fast answer and the categorical fields source, fast reason, and answer form. The question field uses a 1,200-character truncation setting and the complete tokenized input a 512-token limit in both canonical training and main evaluation. Missing average log-probability and logit gap values are replaced by -10 and 0, then clipped to $[-30, 2]$ and $[0, 80]$, respectively. Token count, latency, throughput, and answer length use $\log(1 + x)$; candidate count remains a raw count. Repetition is clipped to $[0, 1]$. Each of the 14 features, including indicators, is standardized using its training-set mean and population standard deviation; standard deviations below 10^{-6} are replaced by 1. These statistics are reused at evaluation. The matched 16-dimensional view uses separate preprocessing, not the canonical feature pipeline.

The matched evidence vector combines six standardized continuous features—average log-probability, logit gap, log token count, log latency, repetition, and log candidate count—with a truncation indicator and nine answer-form indicators. Normalization statistics and the answer-form vocabulary are fitted on training data; source identifiers are excluded.

For $y_f = (w_1, \ldots, w_T)$, average token log-probability is

$$\bar{\ell}_f = \frac{1}{T} \sum_{t=1}^{T} \log p_{\pi_f}(w_t \mid x, w_{<t}), \tag{13}$$

and average logit gap is

$$\bar{m}_f = \frac{1}{T} \sum_{t=1}^{T} \left(z_t^{(1)} - z_t^{(2)} \right), \tag{14}$$

where $z_t^{(1)}$ and $z_t^{(2)}$ are the largest and second-largest logits at step t. Parsing checks an explicit final-answer field, then boxed-answer, GSM-style, and last-answer-like fallbacks. Missing, conflicting, severely repetitive, or truncated outputs receive the corresponding status and format penalty.

Evaluation format penalties. The evaluator assigns $F = 1$ for a recorded generation error, a missing or non-unique candidate, or a status indicating generation-error text, truncation without an answer, no final answer, multiple-answer drift, or degenerate repetition. Otherwise, $F = 0.25$ for a truncated output and $F = 0$ for an untruncated output. These ordered checks define the evaluation penalty. The rescued trainer consumes stored targets and per-example weights; confirming their construction against this evaluator requires the original paired-data records.

A.3 Slow-over-fast utility derivation

If the fast response has already been generated, releasing it gives

$$U_f = A_f - \lambda_{\text{tok}} C_f - \rho F_f. \tag{15}$$

Invoking the slow policy gives

$$U_s = A_s - \lambda_{\text{tok}}(C_f + C_s) - \rho F_s. \tag{16}$$

Table 2: Routing formulations: Q = question, R = fast response, E = fast-side evidence.

Family	Decision	Input	Supervision
Confidence filter	Post-answer	E	None / ranking
RouteLLM-Q	Pre-answer	Q	Model preference
RouteLLM-Q+R(+E)	Post-answer	Q+R(+E)	Model preference
IDPR	Post-answer	Q+R+E	Quality, cost, correction

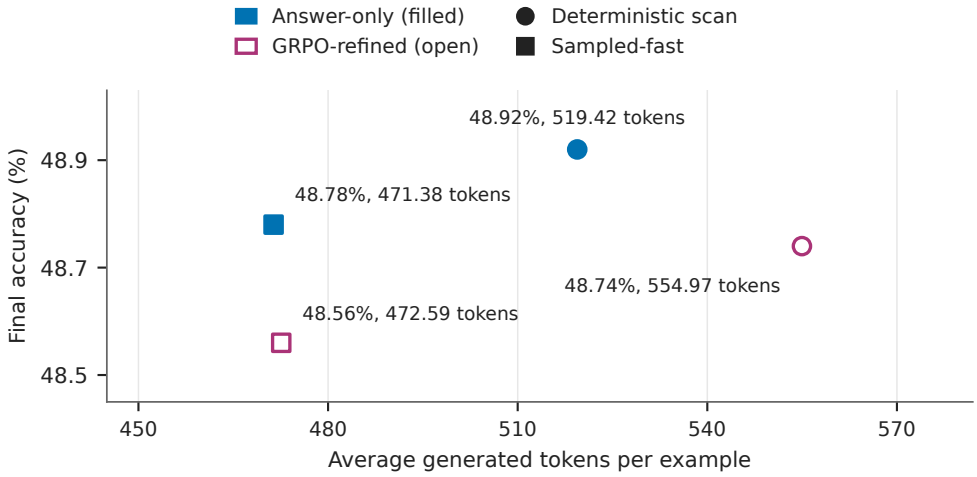


Figure 6: Accuracy and generation cost of four fast-policy/controller configurations. Color and fill identify the fast policy; circles and squares identify deterministic-scan and sampled-fast controllers, respectively. Points are discrete observed configurations, with no connecting curve. Table 6 also reports slow-call rates and gains over each policy's own Always-Fast baseline.

The shared fast cost cancels:

$$U_s - U_f = (A_s - A_f) - \rho(F_s - F_f) - \lambda_{\text{tok}} C_s. \tag{17}$$

The quality and cost targets represent the two components of this difference. Main calibration selects the quality score $\widehat{\Delta Q}$; Equation 10 reports utility and breaks validation accuracy ties.

A.4 Statistical protocol

Accuracy comparisons pair frozen example identifiers. We report percentile confidence intervals for accuracy differences using 5,000 paired bootstrap replicates with seed 20260827. The exact two-sided McNemar test uses discordant correctness outcomes. Holm's step-down correction covers 14 comparisons: main IDPR against Always-Fast, Low Avg LogProb, Q, Q+R, Q+R+E, and IDPR-Matched; each of the four learned routers against Always-Fast; Q+R against Q; Q+R+E against Q+R and Q; and IDPR-Matched against Q+R+E. Table 3 reports seven comparisons.

The intervals are per-comparison bootstrap intervals; the adjusted McNemar tests control family-wise error across the specified comparisons. We use the latter for significance statements. Consequently, a bootstrap interval that excludes zero can coexist with a non-significant adjusted test, as in the comparisons of main IDPR with Low Avg LogProb and IDPR-Matched.

A.5 Routing formulations

Table 2 compares the routing formulations evaluated in this work.

Table 3: Selected paired accuracy comparisons. Accuracy differences and confidence-interval bounds are in percentage points. Bootstrap intervals use 5,000 paired replicates. Exact two-sided McNemar p-values are Holm-adjusted over all 14 comparisons in Appendix A.4.

Comparison	Δ acc.	95% CI	p_{Holm}
IDPR vs. Always-Fast	$+1.02$	$[+0.50, +1.52]$	0.001182
IDPR vs. Low Avg LogProb	$+0.70$	$[+0.16, +1.22]$	0.1064
IDPR vs. RouteLLM-Q	$+1.16$	$[+0.50, +1.82]$	0.004497
IDPR vs. RouteLLM-Q+R	$+1.56$	$[+0.86, +2.24]$	0.0001616
IDPR vs. RouteLLM-Q+R+E	$+2.18$	$[+1.46, +2.88]$	2.624×10^{-8}
IDPR-Matched vs. Q+R+E	$+1.40$	$[+0.80, +2.02]$	0.0001593
IDPR vs. IDPR-Matched	$+0.78$	$[+0.16, +1.40]$	0.1083

Table 4: Quality of selected slow calls. CP and HP are corrective and harmful precision among switched examples, reported as percentages. Net is CP$-$HP, in percentage points.

Method	Selected	CP (%)	HP (%)	Net (pp)
Random Same-k	410	13.41	27.32	-13.91
Low Avg LogProb Same-k	410	21.22	17.32	$+3.90$
Low Logit Gap Same-k	410	11.46	12.20	-0.74
RouteLLM-Q	362	20.99	22.93	-1.94
RouteLLM-Q+R	463	19.22	25.05	-5.83
RouteLLM-Q+R+E	473	16.91	29.18	-12.27
IDPR-Matched	357	23.53	20.17	$+3.36$
IDPR	410	**27.07**	14.63	$\mathbf{+12.44}$

A.6 Controller and fast-policy variants

Figure 6 places the four fast-policy/controller combinations in accuracy–cost space. The deterministic-scan controller is trained on deterministic fast responses paired with rescued slow labels. The sampled-fast variant incorporates sampled fast responses into controller training to expose it to the response variation encountered during policy refinement. Answer-only denotes the main concise-answer fast policy; GRPO-refined denotes a policy-refined fast checkpoint. These configurations vary generation policies and controller training within the same fast-first formulation.

The answer-only fast policy with the deterministic-scan controller has the highest observed final accuracy, 48.92%. With that fast policy, sampled-fast training gives 48.78% accuracy while reducing average tokens from 519.42 to 471.38. The corresponding GRPO-refined configurations reach 48.74% and 48.56%. All four configurations improve on their own Always-Fast baseline. The GRPO-refined deterministic-scan configuration has the largest baseline-relative gain, 1.04 percentage points, but not the highest final accuracy. Table 6 reports the routing gain relative to each fast policy's own baseline; final accuracy and token cost compare the resulting systems.

Additional operating-point comparison. The earlier fixed cost-aware configuration selects 2.10% of test examples and reaches 48.20% accuracy with 143.64 average tokens. The main accuracy-first configuration selects 8.20%, reaches 48.92%, and uses 519.42 tokens. Their corrective precisions are similar (27.62% and 27.07%, respectively): the former uses fewer calls, while the latter gains more accuracy. Because both scores and calibration procedures differ, this compares configurations, not a fixed-score threshold sweep.

A.7 Numerical results underlying the figures

Tables 3–7 preserve the complete numerical summaries for Figures 3–6, including thresholds, call counts, and metrics not shown on the plot axes. Tables 3 and 4 give the data for Figure 3(a) and (b). The figures plot existing summaries without recomputing outcomes or statistical tests.

Table 5: Source-wise results. Accuracy is a percentage; Δ is the gain over the source-specific Always-Fast baseline in percentage points. Tokens are average generated tokens per example. Bold marks each source's highest accuracy, largest gain, and lowest token cost across methods.

Source	Method	Slow rate	Accuracy	Δ	Tokens
GSM8K	Always-Fast	0.00%	60.20	–	**5.31**
	Low Avg LogProb	8.52%	**61.04**	**+0.84**	481.30
	IDPR	6.68%	60.96	+0.76	325.98
MoT	Always-Fast	0.00%	35.60	–	**5.21**
	Low Avg LogProb	7.88%	35.40	−0.20	684.07
	IDPR	9.72%	**36.88**	**+1.28**	712.86

Table 6: Fast-policy and controller variants. Accuracy is a percentage; Δ is the gain in percentage points over the corresponding Always-Fast policy. Tokens are average generated tokens per example. Bold marks the highest accuracy and lowest token cost among the four configurations.

Fast policy	Controller	Slow rate	Accuracy	Δ	Tokens
Answer-only	Deterministic scan	8.20%	**48.92**	+1.02	519.42
Answer-only	Sampled-fast	7.78%	48.78	+0.88	**471.38**
GRPO-refined	Deterministic scan	8.96%	48.74	+1.04	554.97
GRPO-refined	Sampled-fast	7.78%	48.56	+0.86	472.59

Table 7: Calibration and threshold transfer for learned routers. Every method selects 160 of 2,000 calibration examples. Cal. GSM and Cal. MoT count the selected calibration examples by source; Test k counts test slow calls under the unchanged decision rule.

Method	Threshold	Cal. GSM	Cal. MoT	Test k	Test rate
RouteLLM-Q	0.689285	62	98	362	7.24%
RouteLLM-Q+R	0.473572	115	45	463	9.26%
RouteLLM-Q+R+E	0.611396	132	28	473	9.46%
IDPR-Matched	0.7578125[†]	89	71	357	7.14%

[†]Tie-aware unique-identifier boundary rule.